NonStop NET/MASTER Tips and Techniques

by John New

Gresham Software Labs

Email: jnew@greshamsoftwarelabs.com.au

This article originally appeared in *The Connection*, Volume 22, No. 3 - May/June 2001, and is reproduced with permission from the International Tandem Users' Group (ITUG).

Introduction

This is an ongoing column with NonStop NET/MASTER tips and techniques. Each column is also accessible from http://www.greshamsoftwarelabs.com.au/ (Gresham Software Labs). Please send all comments and suggestions to John New at jnew@greshamsoftwarelabs.com.au.

Biography

John New is a technical writer. He has written and updated various Tandem manuals. He currently writes hard-copy, online, and web documents for a variety of software products.

File Handling in NCL: Part 2 – Adding, Reading, Updating, and Deleting Records

This article continues the series on file handling in NCL. Part 1 discussed opening and closing files. This article discusses adding, reading, updating, and deleting records.

Adding a New Record to a File

You can add a new record to entry-sequenced and key-sequenced files. NCL has two statements to add a record—FILE ADD and FILE PUT. For entry-sequenced files, it makes no difference whether you use FILE ADD or FILE PUT. For key-sequenced files, FILE ADD adds a record but FILE PUT is normally used to update an existing record.

When you add a record to an entry-sequenced file, the record is added logically and physically to the end of the file. When you add a record to a key-sequenced file, the record is added logically to the position specified by the value of a record key.

For both entry-sequenced and key-sequenced files, the way you construct a record depends on how the data is physically organized in the file—whether it is mapped, unmapped, or delimited. This in turn determines the form of the FILE ADD or FILE PUT statement.

Regardless of file type and organization, adding a record has two steps: obtaining the record data in one or more variables; and specifying the variable name on the FILE ADD or FILE PUT statement.

Mapped Files

Mapped files can contain variable-length records.

If a file is mapped with the default map ($NCL), then the data for each field is obtained in variables, with one variable representing each field. The FILE ADD statement specifies the variables to construct a record. For example:

```
&var1 = Hello
&var2 = "world!"
FILE ADD VARS=&var*
```

If a file is mapped with a map such as $MSG or $SEC, then you would typically obtain the data by using the MSGREAD or SECALL statement, perhaps do some processing on the data, and then specify the name of the automatically created variable (`&$msg.` or `&$sec.`) in the FILE ADD statement. For example:

```
MSGREAD
/* do some processing */
…
FILE ADD MDO=&$msg.
```

Or:

```
SECALL GET MDO=&$sec.
/* do some processing */
…
FILE ADD MDO=&$sec.
```

Unmapped Files

For unmapped files, it is usual to create fixed length records of fixed length fields before adding a record to a file. For example, the following statements pad the variables `&var1` and `&var2` with spaces so that each variable is ten characters in length:

```
&var1 = LEFT(&var1,10) /* adds five spaces */
&var2 = LEFT(&var2,10) /* adds four spaces */
```

The following FILE ADD statement adds the record to a file:

```
FILE ADD VARS=&var*
```

Delimited Files

Delimited files can contain variable-length records. If the following two assignment statements assign data to the variables `&var1` and `&var2`:

```
&var1 = "Hello"
&var2 = "world!"
```

then the following FILE ADD statement adds the record with the data to a file:

```
FILE ADD VARS=&var*
```

Reading a Record From a File

The FILE GET statement reads records from entry-sequenced, key-sequenced, and edit files. The type of variable you specify in which to place the data from the record depends on how the data is physically organized.

There are three retrieval methods: sequential, exact, and generic.

Sequential Retrieval

You can use sequential retrieval with entry-sequenced, key-sequenced, and edit files. This method retrieves records in a certain direction one at a time, one record for each FILE GET statement.

In an entry-sequenced file, you can retrieve records forward or backward, beginning from the first or last record in a file, respectively. In a key-sequenced file, you can retrieve records forward or backward, beginning from the position specified by the value of a record key. In edit files, you can only retrieve records in a forward direction, beginning from the first record.

The form of the FILE GET statement depends on whether the file is mapped, unmapped, or delimited. For example:

```
/* Mapped with $NCL */
FILE GET OPT=FWD ARGS
/* Mapped with another map */
FILE GET OPT=BWD MDO=&rec.
/* Unmapped */
FILE GET VARS=&field*
/* Delimited */
FILE GET OPT=BWD ARGS
```

Exact Retrieval

Exact retrieval is used only with key-sequenced files. A FILE GET statement specifies a value and begins a search. If the value exactly matches the value of a record, the FILE GET retrieves the record. In the following example, the KEX option requests that only records where the key exactly matches `"SMITH, John"` be retrieved:

```
FILE GET KEY="SMITH, John" OPT=KEX ARGS
```

Generic Retrieval

Generic retrieval is used only with key-sequenced files. A FILE SET statement specifies a value. A FILE GET statement begins the search. If the specified value partially matches the value of one or more records, the FILE GET statement retrieves the first match. Subsequent FILE GET statements retrieve other matches. In the following example, the KEQ option begins a search for records whose key begins with "SMITH". The first match is returned. Subsequent FILE GET statements retrieve other records whose key begins with "SMITH":

```
FILE SET KEY="SMITH"
FILE GET OPT=KEQ ARGS
/* do some processing */

…
FILE GET OPT=KEQ ARGS
```

Updating an Existing Record in a File

The FILE PUT statement updates existing records in key-sequenced files, the information in the new record overwriting the old. You must specify an exact key. In the following example, the information in the variable in `&rec.` replaces the information in the record with the key `"SMITH, John"`.

```
FILE PUT KEY="SMITH, John" MDO=&rec.
```

Deleting a Record From a File

The FILE DEL statement deletes records from key-sequenced files. There are two deletion methods: exact and generic.

Exact Deletion

Exact deletion deletes one record. Using exact deletion, the value of the key you specify must exactly match the value of an existing record key. In the following example, the key must exactly match `"SMITH, John"`:

```
FILE DEL KEY="SMITH, John"
```

Generic Deletion

Generic deletion can delete multiple records. Using generic deletion, the value of the key you specify is regarded as a partial key. All records with a key that partially matches the key you specify are deleted. In the following example, all records with a key beginning with SMITH are deleted:

```
FILE DEL KEQALL="SMITH"
```

Conclusion

As this article shows, adding, reading, updating, and deleting records using NCL is very straightforward.

Later articles in this series on file handling in NCL will discuss topics such as creating user-defined maps with DDL, using file access facilities to communicate with Guardian 90 processes, and using alternate keys in key-sequenced files.